NASA-CR-195857

NAG5-2226

"AMS-ENTERED"

Intelligent Systems and Advanced User Interfaces for Design, Operation, and Maintenance of Command Management Systems

[illegible]

8479
17P

Semi-Annual Report
April 1 - September 30, 1993

William J. Potter, Code 514, Technical Monitor
NASA Goddard Space Flight Center
NAG 5-2226

Christine M. Mitchell
E24-X13
Center for Human-Machine Systems Research
School of Industrial & Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0205
(404) 894-4321
cm@chmsr.gatech.edu

December 1993

(NASA-CR-195857) INTELLIGENT SYSTEMS AND ADVANCED USER INTERFACES FOR DESIGN, OPERATION, AND MAINTENANCE OF COMMAND MANAGEMENT SYSTEMS Semiannual Report, 1 Apr. - 30 Sep. 1993 (Georgia Inst. of Tech.) 17 p

N94-71971

Unclas

Z9/81 0008479

CC: CASI
Tech. officer

Background

Historically command management systems (CMS) have been large and expensive spacecraft-specific software systems that were costly to build, operate and maintain. Current and emerging hardware, software, and user interface technologies may offer an opportunity to facilitate the initial formulation and design of the CMS for specific spacecraft as well as facilitating the training and effectiveness of CMS operations.

Current MOC (Mission Operations Control Center) hardware and software include Unix workstations, the C/C++ programming languages, and an X window interfaces. This configuration provides the power and flexibility to support sophisticated and intelligent user interfaces that exploit state-of-the-art technologies in human-machine interaction, artificial intelligence, and software engineering. One of the goals of this research is to explore the extent to which technologies developed in the research laboratory can be productively applied in complex system such as spacecraft command management. Initial examination of some of the issues in CMS design and operation suggests that application of technologies such as intelligent planning, case-based reasoning, human-machine systems design and analysis tools (e.g., operator and designer models), and human-computer interaction tools, (e.g., graphics, visualization, and animation), may provide significant savings in the design, operation, and maintenance of the CMS for a specific spacecraft as well as continuity for CMS design and development across spacecraft.

Background Analysis

The first six months of this research saw a broad investigation by Georgia Tech researchers into the function, design, and operation of current and planned command management systems at Goddard Space Flight Center. As the first step we attempted to understand the current and anticipated horizons of command management systems at Goddard.

Understanding Current (Representative) CMSs

Command management systems have changed significantly over the past decade. With the advent of powerful and affordable workstations, the CMS system and function can be fully integrated into the mission control center activities, both functionally and physically. Thus, the CMS systems for SAMPEX, WIND, and POLAR, and for all missions henceforth, will be located within the MOC and completely operated by FOT members with mission planning responsibilities.

This realization was important as it curtailed some of the initially planned Georgia Tech activities attempting to understand the history and scope of CMS systems. Initially, we planned detailed analyses of CMSs for such missions as ERBS, GRO, and COBE. Preliminary investigation revealed that the age of these missions meant that the command management function was distributed and a great deal of CMS activity for some of these missions was not conducted within the mission operations facilities (e.g., COBE CMS). Furthermore, some of the CMS functions seemed to be overwhelmed by limitations that current and planned electronic communications would eliminate (e.g., the arduous mission planning process for ERBS).

As a working hypothesis, we decided to assume that the 'workstation CMS' model for form, function and operation, as typified by the SAMPEX CMS, was a good model for current and planned CMS systems. Thus, after spending some time with the mission planning personnel for COBE, ERBS, and GRO, we focused our attention on mission planning operations in SAMPEX, SOHO, WIND/POLAR, with the intention to include EUVE, FAST, and SWAS as time permits.

Detailed Examination of CMS Software

Currently, we have obtained and moved to Georgia Tech two GSFC command management systems, specifically the CMSs for SAMPEX and WIND and POLAR. The SAMPEX CMS offered many advantages. First, since it was written in C++ for a Unix environment, and initially developed on at Sun workstation, we were able, early in this research project, to port the SAMPEX CMS to the Center for Human-Machine System Research at Georgia Tech. Although we have gathered a great deal of documentation on CMSs for many systems, actually being able to run the SAMPEX system and browse the software has provided great insight.

More recently, we obtained a copy of the WIND and POLAR CMS and installed it on our system. Comparison of the two systems demonstrated the lack of commonality that characterizes command management system software. Having both the SAMPEX and WIND/POLAR CMSs available for review provides the Georgia Tech researchers with an important opportunity to compare and contrast two operational systems and to explore the extent and nature of re-use or the lack there-of.

In the next few months, preliminary copies of both the FAST and SOHO CMSs will be available. We plan to extend our current examination of operational and in-development CMSs, by carefully reviewing the resident systems. We will explore the extent to which the FAST CMS, as a SMEX mission with the explicit goal of high re-use, re-uses concepts, functions, code, etc. from SAMPEX. More generally, we will review each new CMS as the various releases become available and understand how they relates to existing (e.g., WIND/POLAR) and other planned (e.g., SOHO) CMSs.

Review of Planned CMS

To ensure that we understand both current (i.e., modern) CMSs and those under development, we have spent a great deal of time trying to learn about CMSs that are currently being developed. We have attended critical design reviews, spoken to CMS developers and FOT responsible for writing requirements, and reviewed documents for both the FAST, SWAS and SOHO command management systems. To ensure that we maintain an accurate and up-to-date perspective, our participation in these processes will continue for the foreseeable future.

Analysis of CMS Operations

In addition to studying the SAMPEX CMS software, we conducted a detailed task analysis of SAMPEX CMS operations. This study documents the operations of SAMPEX mission planning: daily loads, weekend loads, and atypical loads, e.g., 'patch' loads. The goal is to understand operationally what is done, and how. In particular, the study identifies needed inputs to the process (e.g., the RUST, FDF, load requirements), noting the ease with which this information is obtained and entered into the CMS; the CMS load generation process; the CMS load verification process (who does what, e.g., FOT verification, how, with what knowledge); and load uplinking/verification. The study tracks routine operations emphasizing what makes the job 'hard' or cognitively complex, and why.

A report documenting this analysis is in progress. The objective of the study is to understand how mission planning is conducted and the extent to which differences in CMS design affect operations. To ensure generality and highlight differences, a similar study is being conducted of WIND/POLAR CMS operations.

Summary

The next six to twelve months of this project will see the continuation of all of the above activities. As software systems become available, we will obtain copies of the software. By browsing the code, we will continue to explore commonalities and differences, attempting to focus reasons for the lack of extensive re-use and to identify opportunities to facilitate re-use in new systems. We will continue to participate in design reviews by attending the reviews themselves, reading the associated documentation, and interacting with CMS developers and FOT responsible for defining CMS requirements.

Articulation of CMS Commonalties and Causes of Low Re-Use

The next component of this project is an analysis documenting commonalities and differences among CMS that we have studied. The analysis of commonalties and differences together with associated causes of low re-use provides the assumptions that underpin the proposed research and development activities outlined in the section that follows. This analysis is on-going activity, but preliminary findings are summarized below.

The Problem

CMSs are hand-crafted mission by mission at great cost for each individual mission. There is minimal reuse of CMSs from mission to mission. Typically re-use fails to occur across all levels: from conceptual design, to functional specification, to lines of source code.

- o Examination of SAMPEX and the WIND/POLAR CMSs demonstrates that these are very different systems. At the code level the SAMPEX CMS is written in C++ and is object-oriented in design. The WIND/POLAR CMS is written in C and lacks any of the data structures (e.g., objects) that characterize the SAMPEX system.

- o SAMPEX and FAST are both spacecraft in the SMEX mission. One of the SMEX goals is extensive re-use. Comparison of the SAMPEX and FAST CMS (currently only available to GT researchers through examination of the CDR documents) demonstrates that re-use, measured at the lines of code level (i.e., delivered source instruction) is still quite low, approximately 49% re-use for FAST, 31% for SOHO, 43% EUVE.

- o Examination of CMS descriptions, as typified by critical design review materials, demonstrates that even descriptions of CMSs lack a common vocabulary. Comparing and contrasting CMSs, based on available documentation, is very difficult. There is no uniform set of CMS component or CMS functions that characterize how CMS are described and measured.

- o Re-use, when it occurs at all, is measured (and implemented (?)) at the code level--potentially forsaking enhanced productivity and cost savings for re-use at conceptual and functional levels.

- o Because a CMS is essentially hand-crafted for each mission there is little accumulation of experience, either what worked or what did not, from mission to mission.

A Vision of a Solution

The environment in which command management system design occurs is one where multiple design teams independently (and at times concurrently) hand craft from scratch CMSs in response to the needs of space science missions.

Despite the uniqueness of each CMS, they all exhibit a core set of capabilities (e.g., load generation, command database utilization, activity planning, event pool management, activity definition support, etc.). This set of capabilities represents a common and persistent collection of user needs as well as a recurring set of design problems. In many cases, these design problems are resolved independently by the various design teams without regard to prior solutions. Solving recurring design problems in this fashion incurs unnecessary development costs, risks repeating flawed or ineffective designs, and leaves to chance the rediscovery of previously proven design features. It is this type of situation, where the dissemination of past experience could provide great advantage, that makes command management system design an ideal setting in which to explore the utility of case-based reasoning systems.

A case-based reasoning system accumulates experience and makes it available to designers of future systems. Thus one essential component of the proposed solution to facilitate CMS software re-use is use of case-based reasoning technology to accumulate experience and make it available to developers. The section which follows proposes to use case-based reasoning in two ways. The first, a near-term effort, uses a case-based system to make design features of existing CMSs available to CMS developers. The second, a longer-term project, uses case-based reasoning as the knowledge base for a CMS designer's associate. The associate guides CMS designers by suggesting design features from existing CMS applications. When a new design feature is necessary, the associate allows designers to formulate new features as extensions or refinements of existing features. Using a case-based knowledge repository, the associate automatically learns as new features are included in its knowledge base.

In addition to building future command management systems based on the experience of past systems, design would greatly benefit from evolving a common look and feel to command management systems and incorporating standard, commercially available software tools and technologies.

Currently, each CMS specifies and implements its own version of common functions. These functions include interface functionality, data maintenance, and report generation. Consider for example the following command management functions.

The interface software for SOHO is almost 50% of the system. In FAST, 28% of the source code implements interface functions; less than all of this is re-used from existing systems. A common interface across CMSs would facilitate re-use of existing software from conceptual components to actual source code. A common look and feel would also facilitate operator transition from system to system and/or allow the same mission planner to perform the mission planning function for several missions. Informal discussion with FOT suggests that they also would prefer a common look and feel. Currently even with the closely related SMEX missions, FOT staffing plans call for dedicated mission planner for each mission. It is possible that a common look and feel would eliminate the need to have dedicated mission-specific planners.

A common look and feel, if implemented in standard commercial software (e.g., Motif interface library), would begin to evolve a common interface software library. With careful attention to re-use, CMS interfaces would for the most part not only look the same, but the software implementing the interface would for the most part be the same.

Data management and report generation functions show similar problems. For WIND/POLAR, 25% of the software supports data base management and report generation. For SOHO at least 10% is devoted to this function. Yet the functions are essentially the same across all CMS applications. Discussion with designers suggests that a great deal of time and expense is devoted to the development of customized code to maintain data bases and generate required reports Ironically, CMS users are not always pleased with the outcome. For example, the SAMPEX CMS-generated pass plan is discarded and FOT created their own Macintosh-based form. If there was some standardization, a commercial data base product might more cheaply and efficiently accomplish the same functions. Commercial data base report generation capabilities might allow the FOT to design their own reports,
and refine them on an as needed basis.

Finally, initial review of CMS software and associated documentation suggests that a generic CMS software core, similar in concept and form to the TPOCC software, is both feasible and desirable. Generic CMS software would ensure high re-use by ensuring that each new CMS shared common core components, functions, and implementation with all other CMSs. Using the object-oriented metaphor, the CMS generic software would define a CMS class. Each mission would instanciate the generic system, extending or refining it as necessary. As with object-oriented programming, , the common core structure from which all instances (i.e., specific CMS applications) are derived would ensure a great deal of commonality from high-level concepts to the lowest level implementation details. Combined with the case-based designer's associate, described in more detail in the section that follows, a generic CMS core would assure a high degree of commonality due to the core system, and would accumulate design enhancements and extensions to make design experiences, both successful and unsuccessful, available to developers of future systems.

Summary

The preliminary conclusion of this analysis is that extensive re-use can and should be facilitated. Re-use will be greatly facilitated by making previous design experience available to developers via a case-based reasoning system. Development of a common look and feel and the use of standard commercial software will enhance commonalities across systems, and encourage re-use of components developed for one system in subsequent systems. A generic CMS architecture, which each new mission instanciates and extends, will anchor future designs to a common parent. Finally, the case-based designer's associate will guide mission unique extensions and automatically archive new design choices in a development environment which will make those decisions available to designers of future systems.

Project Plan to Facilitate Increased Re-Use

Based on the analysis summarized above, this project proposes a three part approach to facilitate CMS software re-use.

1. **CMS Browser:** A Case-Based Reasoning System to Facilitate Understanding Current CMS Designs

2. **Specification of Generic CMS Core Software**

3. **CMS Design's Associate**

CMS Browser: A Case-Based Reasoning System

The CMS Browser is intended to be the first step in a comprehensive plan to facilitate command management system software re-use. The purpose of the CMS Browser to make knowledge about existing command management systems available to CMS developers.

The CMS Browser is a case-based reasoning system. It will have two major components. The knowledge base is a case base of experience gleaned in the design of existing command management systems including SAMPEX, SOHO, WIND/POLAR, FAST and SWAS. At the level of knowledge/experience contained in critical design review documentation, the CMS Browser will demonstrate the commonalities and unique features of each of these systems.

The second component is the CMS Browser interface. Through its interface the Browser will present to CMS developers a common conceptual model of the components and functions that comprise command management system design. The CMS Browser is hierarchical. It will facilitate the acquisition and efficient maintenance of a conceptual model of a CMS (e.g., activities, triggers) at the highest levels, while making available, via advanced technology tools including visualization, animation, and the case base, successively lower levels of description and mission-specific examples. Successive levels of detail might include trigger types (e.g., event or pass) and actual instances (e.g., cases) showing implementations and associated differences for actual missions (e.g., SAMPEX vs. SWAS vs. FAST). Figure 1 depicts key elements of the CMS Browser.

The goal of the CMS Browser is to foster, i.e., support with effective and intelligent interfaces, the view that CMS are more similar than different. The Browser organizes information around features common to command management systems. The both the interface and structure of the CMS Browser highlight commonalities among CMS implementations. Differences among systems are represented as cases and can be compared and contrasted to explore true differences necessitated by mission requirements versus differences that occur serendipitously--differences that potentially degrade re-use and increase development costs.

Thus, as the first step in facilitating wide-spread re-use of CMS software, the CMS Browser addresses the re-use problem by helping developers acquire a common conceptual model of CMS components and functions. Experience of existing CMS implementations can be viewed as differences or extensions in the context of this common model.

As experience evolves and the knowledge base for the CMS Browser grows, the Browser might turn into a powerful tool through which NASA could monitor re-use and distinguish between necessary and serendipitous differences in proposed command management systems.

The Browser might form the core of a computer-based on-line management and presentation tool through which a proposed CMS design can be described, documented, and presented for review. Each design feature, say at the level of detail of a critical design review, could be structured and presented via the CMS Browser. Each feature could be described either as a re-use of an existing feature or as an extension or addition. Extensions and additions become
new cases in the Browser's case base. The Browser could function as an audit tool with which each 'new case' could be inspected, compared to existing alternatives, and evaluated to ensure that the new feature constitutes a legitimate difference.

The CMS Browser is a first step to facilitate re-use in that it makes knowledge about existing CMS design readily available. The CMS Browser, used as a on-line documentation and presentation tool, encourages re-use by isolating, inspecting and evaluating each mission unique feature, i.e., a feature not exhibiting re-use. The next steps in this process, however, support re-use by providing a core set of generic CMS modules and a development environment through which mission-unique requirements are specified, and recorded for future use in a case base.

A Generic CMS Software Core

Two experiences regarding software re-use at Goddard Space Flight Center are very instructive. The first concerns the SAMPEX CMS. Although developed as a proof-of-concept demonstration system, the SAMPEX CMS had surprising little impact on future generations of command management systems. Apparently neither the software itself nor its associated documentation had substantial impact in facilitating re-use of many of its many innovative concepts and design features. As a design artifact, the SAMPEX CMS might be considered an example . The SAMPEX experience suggests that using examples to facilitate re-use may not be an effective strategy.

The second experience is the TPOCC software. Widely regarded as a success, the TPOCC software provides each mission with core capabilities, functions, and code. Mission-unique features are added only when the core system fails to provide necessary functionality. Using the object-oriented paradigm of generic structure and function which is instanciated and extended, the TPOCC software facilitates widespread re-use.

As a paradigm for re-use, the TPOCC model of defining a generic core system is much more promising than the SAMPEX strategy of providing an insightful example. Thus, this project proposes the specification of a generic CMS software architecture from which future command management systems can be built.

The next step in the development of a generic core is collaboration with CMS developers to begin to define the components and functions that comprise such a core. To ensure applicability, it might be advisable to develop the generic core in conjunction with the development of a mission-specific CMS, delineating at every step core components from mission-specific extensions.

The CMS generic core system will help to ensure re-use by establishing a collection of components and functions common to all mission. Design re-use due to learning from previous experience complements re-use attributable to the core. Re-use based on learning from past experience can only occur when existing design experience is accessible. Thus, the final component of the proposed project to facilitate CMS software re-use is the development of a CMS designer's associate that makes previous experience accessible and encapsulates the experience of new design choices.

The CMS Designer's Associate

This proposal suggests a framework for a computer-based designer's associate (DA) which supports design efforts in which experience determines, more than any other factor, designer performance. Figure 2 illustrates the DA framework.

Within this framework, the DA augments the designer's experience; it does not replace the designer nor subsume any normal duties. It is intended only to extend the designer's reach. Humans are good at creative adaptation but poor at remembering a full range of design cases because they tend to be biased in their remembering. On the other hand, those that lack relevant experience may not have sufficient knowledge to solve the design problem effectively. The DA can augment the memory limitations of humans, providing them with design cases they would otherwise fail to remember. The DA framework attempts to use the best qualities of both human and computer for solving design problems.

The DA framework has several facets: a domain specific conceptual framework, a design ontology, the DA experience-base, a design editor, and the designer's associate engine (i.e., the DA). The conceptual framework is used to organize and index design experience and may be seen as both a standard vocabulary and domain taxonomy for describing goals, needs, concepts, and the like within the domain of interest (e.g., the concepts of activity and activity dictionary are important concepts in command management).

The design ontology describes the nature of the design process and the design product in a domain and problem independent fashion. It allows one to capture the iterative and incremental character of the design process as well as the interdependent quality decisions about the problem context and its solutions. Both the design product (i.e., the product or output of the design process--e.g., a mission-specific command management system) and the DA experience base is defined in terms of the design ontology. Thus, expanding the DA experience base is a matter of merging the existing base with existing or past design products.

The design editor (a user interface) is the means through which the designer formulates his/her designs, views those of others, and interacts with the DA. The editor utilizes the conceptual framework to assist the designer in forming queries concerning past designs, organizing problem context descriptions and design decisions, and formulating plans. This architecture allows the editor and DA to be specialized by using a domain specific conceptual framework. The designer's associate is composed of two components: a designer's associate engine built upon a case-based reasoner and the experience base. This architecture allows the DA to be further specialized by using problem specific experience bases.

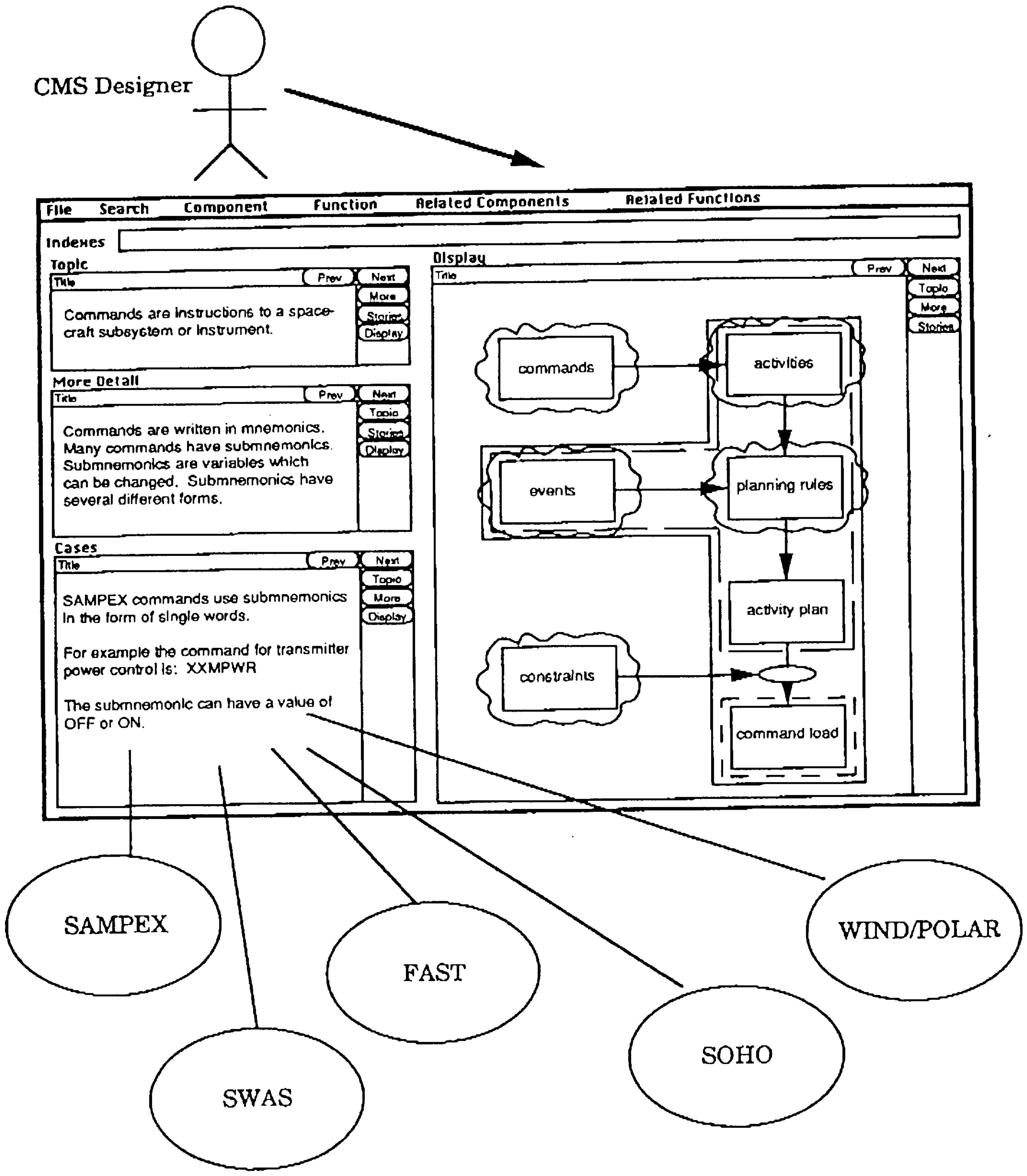
For our purposes, a DA is the junior partner of a design team consisting of a human designer and the computer-based DA. The role of the DA is to recall past design decisions or experiences whose problem context or solution is similar to that being considered by the designer. DA decision making is limited to determining the relevance of past design experiences in light of the designer's current goals (i.e., that set of issues and requirements that are the current focus of the designer's efforts). The role of designer is to create solutions to new problems and re-use or adapt solutions to re-recurring problems based on design experiences recalled by the DA as well as those with which the designer is personally acquainted.

The DA expands its experience base through its interaction with the designer by remembering their decisions. The objective of the DA framework is to facilitate the collection and dissemination of design experience. Design re-use can clearly improve designer performance over that which can be achieved when re-use is not considered. That is, both designer productivity and design quality may be enhanced. Productivity is potentially enhanced because less effort is involved in assimilating design knowledge (if properly represented) than recreating it. Quality is often enhanced though the design equivalent of natural selection. Over time, good designs are re-used, are adapted, or guide and therefore become dominate while bad designs are discarded relatively quickly. Yet, design re-use is not common.

Despite its merits, design re-use as well as the utilization of the underlying experience has been hindered for want of a formal mechanism for disseminating relevant design information throughout the CMS design community. Re-use has also been hindered by the lack of a common conceptual framework. Without such a framework there is no widely accepted and understood vocabulary for describing design goals, user requirements, or design solutions. Differences in vocabulary not only induce design variability, but also obscure design similarity. The DA framework is intended to address both of these issues through facilitating the collection and dissemination of design knowledge in the form of past design experience and fostering the growth of a common conceptual framework within the CMS design community. By serving as a focal point for the collection and dissemination of design experience based upon a common conceptual framework the DA will foster the implicit collaboration of CMS designers in a manner that transcends both temporal and project boundaries.

CMS Browser:

A Case-based Reasoning System



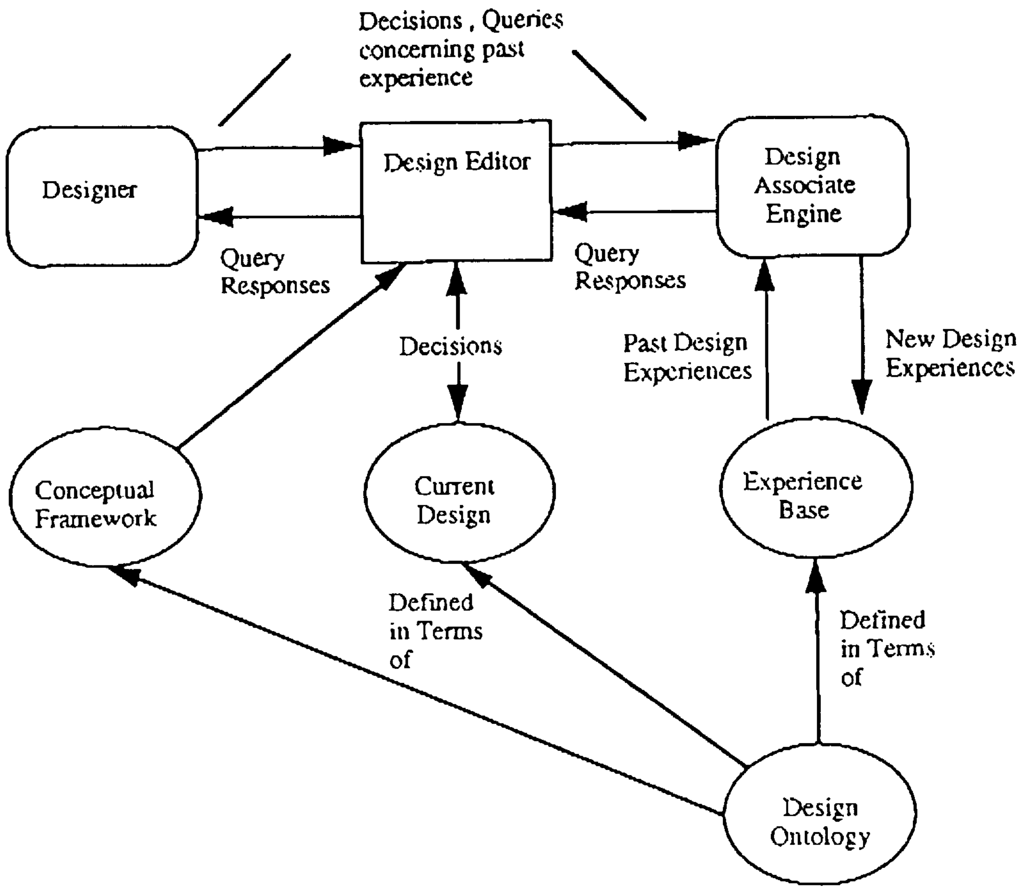
Commands are instructions to a space-craft subsystem or instrument.

Commands are written in mnemonics. Many commands have submnemonics. Submnemonics are variables which can be changed. Submnemonics have several different forms.

SAMPEX commands use submnemonics in the form of single words.

For example the command for transmitter power control is: XXMPWR

The submnemonic can have a value of OFF or ON.


Decisions , Queries concerning past experience
Designer
Design Editor
Design Associate Engine
Query Responses
Query Responses
Decisions
Past Design Experiences
New Design Experiences
Conceptual Framework
Current Design
Experience Base
Defined in Terms of
Defined in Terms of
Design Ontology

Understanding Command Management Systems: How they operate. How they are developed

- Port CMS applications to Georgia Tech
  - SAMPEX CMS
  - WIND/POLAR
  - FAST (as available)
  - SOHO (as available)

- Understand Commonalities and Differences Across CMSs
  - Attend CMS Design Reviews
  - Review CMS Design Documents (SAMPEX, WIND/POLAR, SOHO)
  - Interviews with CMS Developers
  - Interviews with FOT responsible for developing requirements

- Conduct Task Analysis of CMS Operations
  - SAMPEX
  - WIND/POLAR

Articulation of CMS Commonalities and Causes of Low Re-Use

- CMSs are more similar than different.
- Low re-use in part stems from failure to standardize on common components.
- Low re-use in part stems from a lack of availability experience/information of previous designs.

Project Plan to Facilitate Increased Re-Use

Assumptions

- Re-use would be facilitated by evolving a corporate memory of existing systems that is easily accessible to developers of new systems.

- Re-use would be facilitated by defining a common CMS core software system, like TPOCC software, such that each mission would instanciate (i.e., refine and extend) the common core.

- Re-use would be facilitated by providing a developer with a designer's associate through which the designer would instanciate mission-specific features of the core system. The case base would facilitate re-use of existing design concepts and components. As necessary, new design features, i.e., features that are added or extensions of existing features, are specified via the associate's development environment and automatically added to the associate's case base.

Activities

1. **CMS Browser: A Case-Based Reasoning System to Facilitate Understanding Current CMS Designs**

2. **Specification of Generic CMS Core Software**
   * **intended as part of TPOCC software**
   * **mission-specific CMS as an instanciation of core system**
   * **use commercial-of-the-shelf software (e.g., editors, interface widgets, data base systems, etc.)**
   * **evolve a common look and feel**

3. CMS Design's Associate: A Case-Based Design Environment to Specify New CMS Applications Archive Design Decisions